Project Phase III Report

[redacted], [redacted], Matthew Lieb, [redacted]

Professor Dailey

ME 321 - Introduction to Heat Transfer

7 May 2021

Table of Contents

Abstract

This project seeks to model all of the components of heat gain and heat loss impacting Professor Dailey's in-ground pool to predict if the pool temperature is suitable for swimming, around 85 degrees Celsius, by May 7th. The problem is modeled in MatLab using the air temperature as an input to find the heat gain in the pool throughout the day. Heat transfer in the pool system occurs through modes including radiation, forced convection, and conduction between the concrete and the surroundings. Based on the results of the MatLab simulation, a natural gas heater will be needed if the pool temperature is to reach 85 degrees Celsius by May 7th.

Introduction

Professor Dailey's outdoor pool is an 18-foot by 42-foot rectangular pool with a depth ranging from 1-foot in the shallow end to 8.5-feet in the deep end. Within the pool is a large sundeck 12-feet by 7-feet of constant 1-foot depth. The pool walls are made of concrete with a minimum 6-inch thickness. Additionally, the pool is covered at almost all hours of the day by a solar blanket on the surface of the water intended to minimize evaporative cooling and evaporative losses. Given these parameters, all forms of heat transfer between the pool and the surroundings can be modeled to determine if the pool will reach a suitable temperature for swimming by a given day. There are several modes of heat transfer between Professor Dailey's pool and the surroundings. Conduction, radiation, and convection all appear in at least one form. In terms of conduction, heat is lost from the pool water through the surrounding concrete walls and floor. This occurs because the ground temperature is cooler than the water temperature. In terms of convection, heat loss will occur by free convection between the pool water and the pool floor, walls, and surface. The walls of the pool, including the walls that compose the sundeck, were modeled as vertical plates. The floor of the pool was modeled as an inclined plate as it is sloped due to the changing pool depth. Convection was also considered for the pool-air interface and was modeled as a horizontal plate. In terms of radiation, heat is gained through incident solar irradiation from the sun. Additionally, radiative exchange occurs between the high atmosphere and the pool surface.

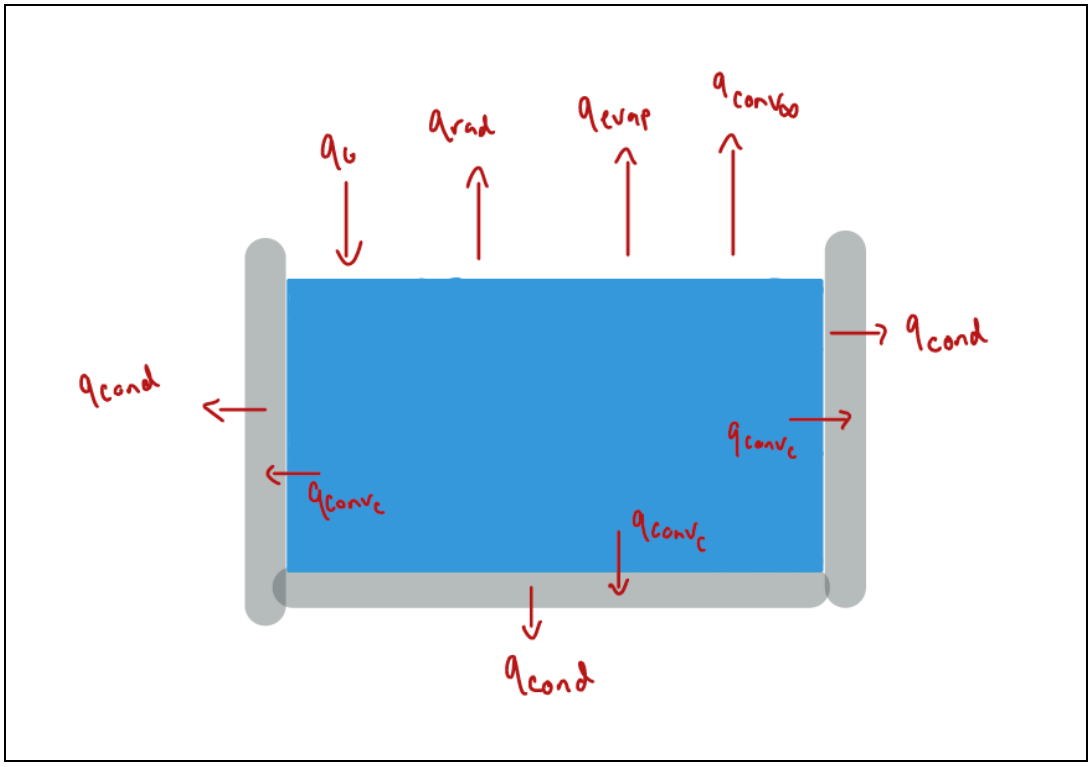

Figure 1: Pool Thermal System Schematic

Methodology

The modes of heat transfer used to model Professor Dailey's pool include conduction, convection, and radiation. Conduction occurs between the concrete pool walls and floor and the surrounding soil. This is modeled using Fourier's law which can be found in the conduction equations below. Convection occurs between the pool surface and the surrounding air, as well as between the pool water and walls, and floor. All of the convection was modeled as free convection because there is no forcing velocity, rather buoyancy forces are driving the heat transfer. The pool walls were modeled as vertical plates, the surface of the pool was modeled as a horizontal plate, and the floor of the pool was modeled as an inclined plane. Radiation occurs as incident solar radiation from the sun as well as radiation between the pool surface and the atmosphere. To model solar radiation, the Bird Clear Sky Model was implemented in MatLab. Radiation between the pool surface and the atmosphere was modeled using the modified Stefan-Boltzmann equation.

Assumptions:

- 42’ long pool walls were treated identically despite differences between the walls due to the sundeck
- To model the pool walls as vertical plates, the average pool depth was used for the length
- The pool bottom was modeled as a constant decreasing slope of 7.82deg whereas it is different in separate sections of the pool
- Assumed that temperature of concrete equals that of the pool water
- Evaporative losses and cooling are negligible due to the solar blanket

Conduction Equations:

Conduction between the pool walls and floor, and the surroundings

- $q_{cond} = -kA\frac{dT}{dx}$ (1)

Radiation Equations:

Radiative exchange between pool and atmosphere:

- $q_{rad} = \varepsilon\sigma A(T_s^4 - T_{surr}^4)$ (2)

Solar irradiation:

- Modeled using Bird Clear Sky Model (G(hour))
- $q_{rad} = \alpha G(hour)A_S$ (3)

Convection Equations:

Volumetric thermal expansion coefficient:

- $\beta = 1/T$ (for ideal gasses) (4)
- Tables A.5, A.6 (for liquids)

Film temperature:

- All properties evaluated at film temperature, T_f
- $T_f = \frac{(Ts + T\infty)}{2}$ (5)

Rayleigh number:

- $Ra_L = \frac{g\beta(Ts - T\infty)x^3}{\nu\alpha}$ (6)

Free convection vertical plates:

- (9.26) $\overline{N_{UL}} = \{0.825 + \frac{0.387Ra^{1/6}}{[1+(0.492/Pr)^{9/16}]^{8/27}}\}^2$ (all conditions)

(7)Free convection horizontal plates:

- (9.30) $\overline{N_{UL}} = 0.54Ra_L^{1/3}$ $(10^4 < Ra_L < 10^7)$ (8)
- (9.31) $\overline{N_{UL}} = 0.15\, Ra_L^{1/3}$ $(10^7 < Ra_L < 10^{11};\ all\ Pr)$ (9)
- (9.32) $\overline{N_{UL}} = 0.52\, Ra_L^{1/5}$ $(10^4 < Ra_L < 10^9;\ Pr > 0.7)$ (10)

Free convection inclined plates:

- In Rayleigh number, replace g with gcos(Θ) for $0 \leq \Theta \leq 60$ degrees
- (9.26) $\overline{N_{UL}} = \{0.825 + \frac{0.387Ra^{1/6}}{[1+(0.492/Pr)^{9/16}]^{8/27}}\}^2$ (all conditions) (11)

Heat transfer coefficient:

- $\overline{h} = \frac{N_{UL}k_f}{L}$ (12)

Heat transfer:

- $q_{conv} = hA(T_s - T_\infty)$ (13)

Constants:

Quantity	**Initial guess**	**Ranges for Sensitivity Analysis**	**Final Value**
g (gravity)	$9.81\ m/s^2$	$9.81\ m/s^2$	$9.81\ m/s^2$
σ (Stefan-Boltzmann constant)	$5.67 * 10^{-8}\ W/m^2 \cdot K^4$	$5.67 * 10^{-8}\ W/m^2 \cdot K^4$	$5.67 * 10^{-8}\ W/m^2 \cdot K^4$
ε_{water} (water emissivity – section 12.9.3)	0.97	0.97	0.97
α (surface absorptivity of water)	1	N/A - Not included in Phase II model	1
V (volume)	$105.85\ m^3$	$100 - 110\ m^3$	$105.85\ m^3$
A_S (surface area of pool)	$69.915\ m^2$	$65 - 75\ m^2$	$69.915\ m^2$
ϱ_{water} (density of water)	$1000\ kg/m^3$	$1000\ kg/m^3$	$1000\ kg/m^3$
c_p (heat capacity of water)	$4184\ J/kg \cdot K$	$4127 - 4195\ J/kg \cdot K$	Function of temperature - Interpolated
T_{soil} (temperature of soil)	$283\ K$	N/A - Not included in Phase II model	$283\ K$
T_{air}	300 K	300 - 350 K	Input data
k_{soil} (heat transfer coefficient for red clay loam)	$0.72\ W/m \cdot K$	N/A - Not included in Phase II model	$0.78\ W/m \cdot K$
u (wind velocity)	0.1 m/s	0.1 - 30 m/s	0.1 m/s
ṁ	3.86e-04 *kg/s*	3.86e-04 - 1.93e-03 *kg/s*	3.86e-04 *kg/s*

Table 1: Parameters

Results

For Phase I of the experiment, we treated the pool as a control mass and performed an energy balance on it to develop our model. The first estimation included simplified variables and parameters such as constant air temperature, wall temperature, and thermodynamic parameters for air and water. Additionally, simplifying the pool geometry offered a more straightforward convection coefficient and allowed us to create our working model. The working model was run for a single day with considerations for the solar cycle. The results show a relatively linear increase in water temperature from 10.2 degrees to 14.7 degrees Celsius. Even after changing the constant air temperature to 6-hour air temperature intervals, the results did not conform to any expectations.

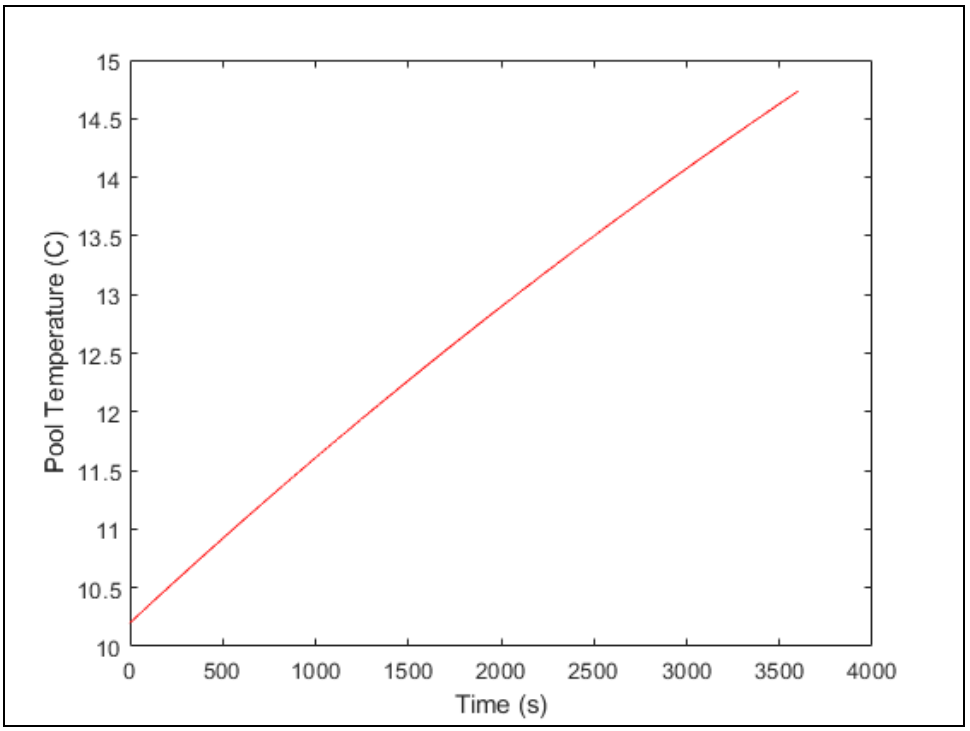

Figure 2: Phase 1 Model Prediction (24hrs)

The Phase II model saw improvement graphically, having a more parabolic function over a 24 hour period. Changing air constants from property values at 300K to 350K resulted in a 25% increase in head gain for our model, showing a high sensitivity to air temperature. Accounting for this, an interpolation function was added in order to use specific air values in the model. Another significant variable that showed a high sensitivity in our results was the Prandtl number of the pool water. To account for this, another interpolation function was added for the Prandtl number and the thermal expansion coefficient. A factor that proved to be less significant than anticipated was the effects of evaporation. Due to the use of the solar blanket as well as the negligible effect that the mass flow rate of air had on final water temperature, this parameter was left as a constant in the model. Variables not yet accounted for were the incident solar radiation effects and conduction between the concrete soil and the underground soil, both of which were assumed to have an effect on the final water temperature.

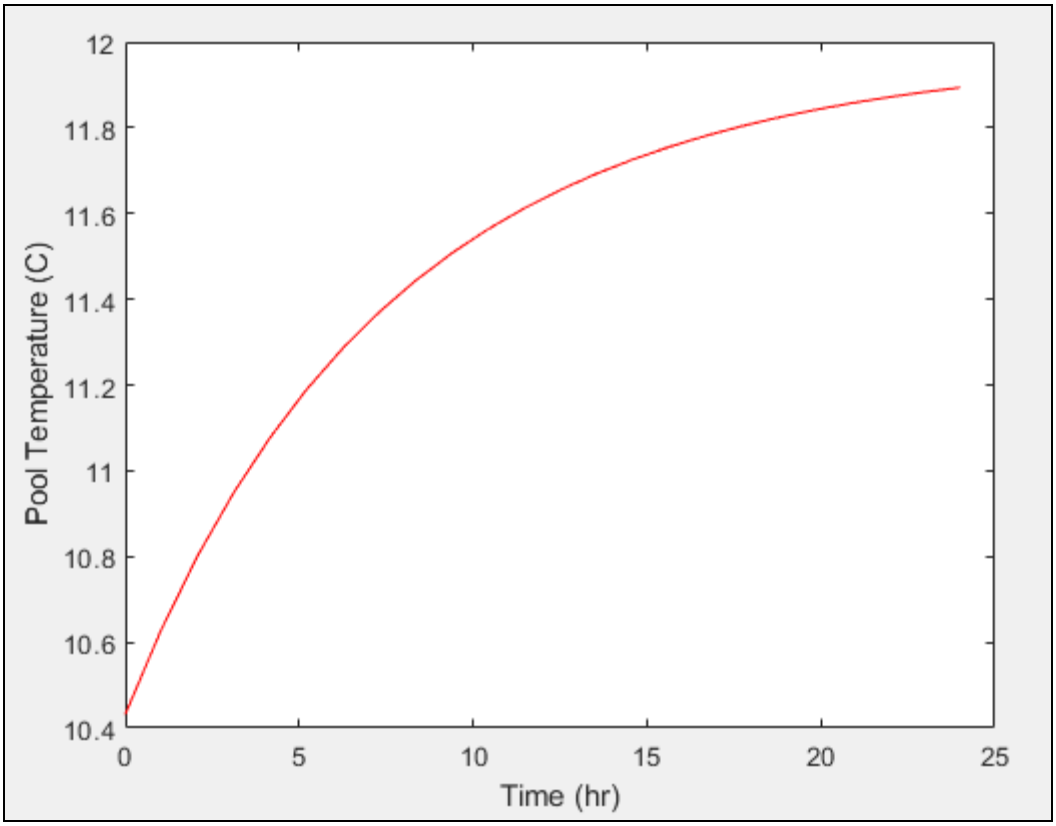

Figure 3: Phase II Model Prediction (24hrs)

In the final phase, there were still significant aspects of the energy balance equation to add to the model. This included more variable air temperatures, conduction between the pool walls and the underground soil, and incident solar irradiation. The added accuracy in air temperatures allowed us to finalize the air aspect of the model. Conduction between the concrete and soil added a cooling element to the model, further adding accuracy to our results. Incident solar irradiation was a crucial aspect of the model and was the most significant heating element in our calculations. With these components added, the model produced promising results. Upon further inspection of the model, a few stand-out variables affecting a significant portion of the result were questionable. Effective sky temperature was among the list that most influenced the final results. After debugging and finalizing the changes, the model produced agreeable results.

Figure 4 shows a 24-hour cycle of the pool temperature starting at 9 A.M. Using the Bird Sky Model, the irradiation "turns on" from 10 A.M to 5 P.M., hence the large increase in temperature during those times. Based on our results in Figure 5, we recommend heating the pool with a solar blanket to help it reach 85 degrees Fahrenheit (29.4 degrees Celsius). A 6-day cycle shows that we estimate the final temperature after 6 days to be 16.7 degrees Celsius.

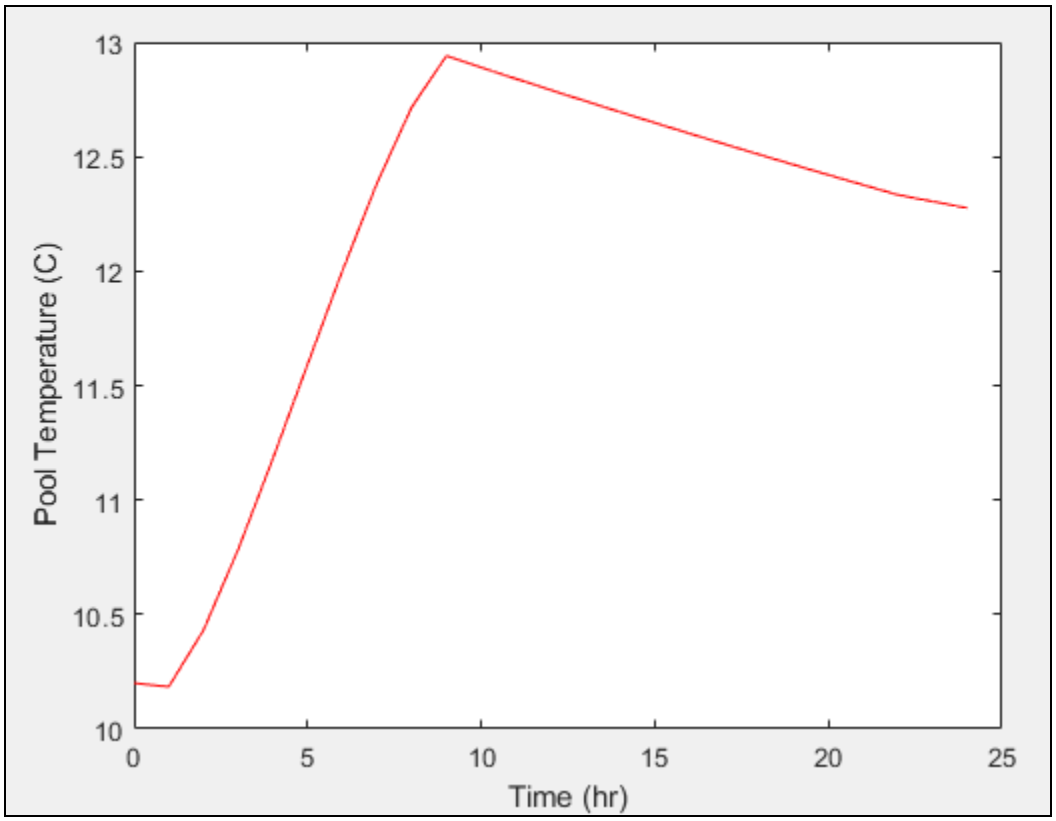

Figure 4: 24 Hour Temperature Cycle

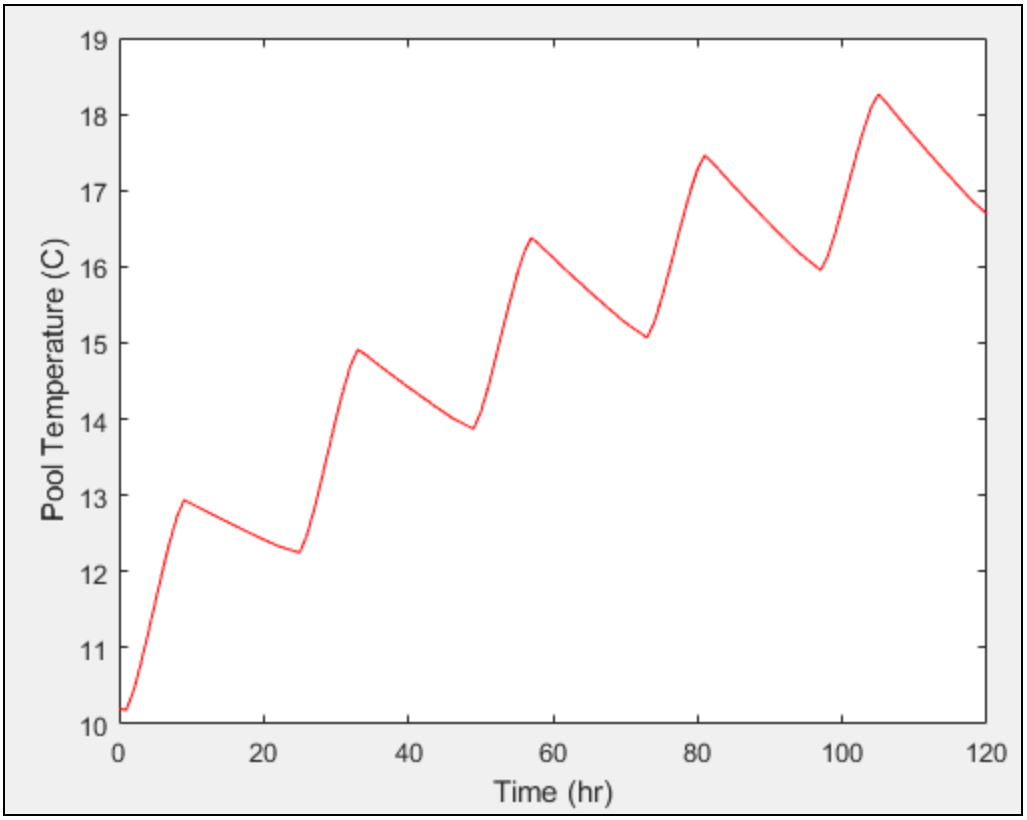

Figure 5: 6 Day Temperature Cycle

Discussion

The final model prediction is on target given the data from 4/26/21. We made a number of simplifications to best approximate real-world physics, with some approximations holding more weight than others. For instance, we modeled the pool using vertical and horizontal plates experiencing free and forced convection. This approach allowed us to get a working model and after performing our sensitivity analysis, we found that convection had a relatively low effect on heat loss. Other phenomena such as conduction, variable air temperatures, and thermophysical properties, and solar irradiation were very important and required fine-tuning in Phase III. One of the biggest sources of deviation in our results from the measured data is solar irradiation. We

used the Bird Sky Model that “turns on” after 1 hour and “turns off” after 7 hours, with no irradiation at any other time. Obviously, this is not how real-world physics works, but this model accounts for latitude, longitude, time zone, and pressure to calculate an estimate for irradiation. As such, we believe that the model approximating real-world physics and its implementation using discrete values at certain time intervals is a big contributor to differences in our results versus the measured data. Another source of error is radiative losses. This quantity was calculated using approximated effective sky temperatures. Effective sky temperatures range from 230 K to 285 K. But we assumed a uniform daytime and nighttime sky temperature of 5 and 10 degrees Celsius, respectively. Lastly, conduction was also an important mode of heat transfer that was the primary source of heat loss. We assumed that the concrete lost heat by conducting energy through a 6-inch wall all around the pool. We set the pool wall temperature equal to the water temperature and assumed that the temperature of the other side of the concrete wall was equal to a uniform soil temperature of 9.85 degrees Celsius. We saw that conduction was one of the largest sources of heat loss in our model and likely made a big contribution to the deviation from measured pool temperature data.

Conclusions

This project intends to model all of the modes of heat transfer that occur between a pool and its surroundings to determine its final temperature by a certain day and make a recommendation regarding whether or not a gas heater needs to be used to further heat the pool. To answer this question of whether or not the pool is warm enough, all of the modes of heat transfer between the pool and the surroundings were modeled in MatLab and set up so that the pool temperature after a heating cycle is output. Key concepts including conduction, radiation, and convection are integrated into the model and the final result reveals that a gas heater is needed to further warm the pool so that it is suitable for swimming by May 7, 2021. We believe that solar irradiation and conduction were two of the most important sources of deviation in our model. We estimate a final temperature of 16.7 degrees Celsius after 6 days of simulation.

References

Karn, A., Chintala, V., & Kumar, S. (2019, April 08). An investigation into sky temperature estimation, its variation, and significance in heat transfer calculations of solar cookers. Retrieved May 07, 2021, from https://onlinelibrary.wiley.com/doi/full/10.1002/htj.21459

Bird clear sky model. (n.d.). Retrieved May 07, 2021, from https://www.nrel.gov/grid/solar-resource/clear-sky.html

Contribution Statement

[redacted] – MatLab programming and debugging, thermal modeling, parameter calculations, sensitivity analysis calculations, contributed to writing reports.

[redacted] – MatLab programming for phase I,II,III, added time adjustment for graphs, bird sky function for irradiation, temperature loops as well as interpolation functions for various values from the textbook. Additionally added conduction and radiation + irradiation components for heat transfer

Matthew Lieb – Modeled pool volume, conduction and radiation effects on heat loss/gain for the pool system. Contributed to write-ups in all phases of the model.

[redacted] – Contributed to Phase 1 energy balance. Contributed to Phase 2 and final write up.

Appendices

MatLab code:

```
clear; close; clc;

%% Constants
v = 105.85;                 % [m^3] re-evaluate once given stair width, neglects corner radius
areaSurface = 69.915;       % [m^2]
sigma = 5.67E-8;            % [W/m^2*K^4]
g = 9.81;                   % [m/s^2]
area1 = 18*.0929; %Areas for various faces of the pool
area2 = 214*.0929;
area3 = 153*.0929;
area4 = 197.25*.0929;
area5 = 35.5*.0929;
area6 = 19*.0929;


%% Parameters
%Start Time
timeStart = 9; %CHANGE THIS THIS IS THE STARTING TIME FOR
timeStart = round(timeStart);

% Water
tempWater = 10.2; %[c] %CHANGE THIS, THIS IS THE STARTING POOL TEMP
tempWater = tempWater + 273; %[k]
tempStart = tempWater;
rhoWater =  1000;                   % [kg/m^3] %Effectively Constant
cpWater = 4184;                     % [J/kg*K] %Effectively Constant
latentHeatWater = 2453.5;     % [KJ/kg]
muWaterArray = [1850E-6 1652E-6 1422E-6 1225E-6 1080E-6 959E-6 855E-6 769E-6 695E-6
631E-6 577E-6 528E-6];              % [N*s/m^2]
kWaterArray = [569E-3 574E-3 582E-3 590E-3 598E-3 606E-3 613E-3 620E-3 628E-3 634E-3
640E-3 645E-3];                     % [W/m*K] (@ 20 degC water, not film)
betaWaterArray = [-68.05E-6 -32.74E-6 46.04E-6 114.1E-6 174E-6 227.5E-6 276.1E-6 320.6E-6
361.9E-6 400.4E-6 436.7E-6 471.2E-6];                   % [K^-1] lots of variation 20C
prWaterArray = [12.99 1.22 10.26 8.81 7.56 6.62 5.83 5.20 4.62 4.16 3.77 3.42];

epsilonWater = 0.97;                    % section 12.9.3
```

```
% Air
tempAirArray = [23,19,9,17,16];  %[c] %CHANGE THIS INPUT EACH TEMPERATURE
FOR EVERY DAY
temoAirArray = tempAirArray + 273; %[K]
cpAirArray = [1.006E3 1.007E3 1.009E3];                    % [J/kg*K] %250 300 350
nuAirArray = [11.44E-6 15.89E-6 20.92E-6];                 % [m^2/s]
kAirArray = [22.3E-3 26.3E-3 30E-3];                  % [W/m*K]
alphaAirArray = [15.9E-6 22.5E-6 29.9E-6];              % [m^2/s]
prAirArray = [.720 .707 .700];

%Soil- High Clay Composition(Red Clay Loam) - Higher end
tempSoil = 283; %[K]
ksoil = .78; %[W/m*K]

% Walls and bottom
areaWall = 57.390;              % [m^2]
kConcrete = 1.4;                % [W/m/K]
lConcrete = 6*.0254;            % [m]
areaBottom = areaSurface; % [m^2]
tempWall = tempWater;

% Evaporation
pw = 2.34E-3;                   % [Pa] saturated vapor pressure
pa = 0.94E-3;                   % [Pa] saturation pressure at air dew point
u = 0.1;                        % [m/s] air velocity
mDot = areaSurface*(pw-pa)*100*(0.089+0.0782*u)/(latentHeatWater);      % [kg/s]


%Effective Sky Temperature ranges from  230K to 285k (cold clear, to warm
%overcast) - averages at 0 deg c or 273K
%nighttime sky temp uniform around roughly 5 deg c (higher than daytime)
%daytime 5deg c, night 10deg c
tempSkyArray = [5 + 273, 10 + 273]; %[k] Day and Night
cover = 1; %Cover is on if 1 0 if off
tempSky = tempSkyArray(1);
G = [0
    0
    0
    0
    0
```

```
    0
    0
    0
    0
    465
    658
    745
    780
    781
    748
    665
    481
    0
    0
    0
    0
    0
    0
    0];
absorptivity = 1; %Surface Absorptivity of Water

t = 0:1:length(tempAirArray)*86400; %Number of values in air temperature arrays used for how
many days the loop goes for
T = zeros(length(tempAirArray),1);
deltat = 1;
T(1,1) = tempWater;
day = 1;
time = timeStart;

for i=2:length(t)
    tempAir = tempAirArray(day);
    cpAir = 0;
    nuAir = 0;
    kAir = 0;
    alphaAir = 0;
    prAir = 0;
    betaAir = 1/tempAir;
    if tempAir < 300 %Interpolation functions depending on temperature range (3 values so 2
ranges)
        cpAir = cpAirArray(1) +(tempAir-250)*(cpAirArray(2) - cpAirArray(1))/(300-250);
```

```
        nuAir = nuAirArray(1) +(tempAir-250)*(nuAirArray(2) - nuAirArray(1))/(300-250);
        kAir = kAirArray(1) +(tempAir-250)*(kAirArray(2) - kAirArray(1))/(300-250);
        alphaAir = alphaAirArray(1) +(tempAir-250)*(alphaAirArray(2) -
alphaAirArray(1))/(300-250);
        prAir = prAirArray(1) +(tempAir-250)*(prAirArray(2) - prAirArray(1))/(300-250);
    else
        cpAir = cpAirArray(2) +(tempAir-300)*(cpAirArray(3) - cpAirArray(2))/(300-250);
        nuAir = nuAirArray(2) +(tempAir-300)*(nuAirArray(3) - nuAirArray(2))/(300-250);
        kAir = kAirArray(2) +(tempAir-300)*(kAirArray(3) - kAirArray(2))/(300-250);
        alphaAir = alphaAirArray(2) +(tempAir-300)*(alphaAirArray(3) -
alphaAirArray(2))/(300-250);
        prAir = prAirArray(2) +(tempAir-300)*(prAirArray(3) - prAirArray(2))/(300-250);
    end

    muWater = 0;
    kWater = 0;
    prWater = 0;
    betaWater = 0;
    tempWaterArray = [273.15 275 280 285 290 295 300 305 310 315 320 325];
    %No need for film temp as edge of concrete is close to water temp
    n = 0;
    if tempWater <= 275 %Finding ranges for temperatures for interpolation functions
        n = 1;
    elseif 275 < tempWater <= 280
        n = 2;
    elseif 280 < tempWater <= 285
        n = 3;
    elseif 290 < tempWater <= 295
        n = 4;
    elseif 295 < tempWater <= 300
        n = 5;
    elseif 300 < tempWater <= 305
        n = 6;
    elseif 305 < tempWater <= 310
        n = 7;
    elseif 310 < tempWater <= 315
        n = 8;
    elseif 320 < tempWater <= 325
        n = 9;
    end
```

```
   muWater = muWaterArray(n) +(tempWater - tempWaterArray(n))*(muWaterArray(n+1) -
muWaterArray(n))/(tempWaterArray(n+1) - tempWaterArray(n)); %Interpolation Functions
   kWater = kWaterArray(n) +(tempWater - tempWaterArray(n))*(kWaterArray(n+1) -
kWaterArray(n))/(tempWaterArray(n+1) - tempWaterArray(n));
   prWater = prWaterArray(n) +(tempWater - tempWaterArray(n))*(prWaterArray(n+1) -
prWaterArray(n))/(tempWaterArray(n+1) - tempWaterArray(n));
   betaWater = betaWaterArray(n) +(tempWater - tempWaterArray(n))*(betaWaterArray(n+1) -
betaWaterArray(n))/(tempWaterArray(n+1) - tempWaterArray(n));
   alphaWater = kWater/(rhoWater*cpWater);   % [m^2/s]
   nuWater = muWater/rhoWater;          % [m^2/s]

   raWall1 = g*betaWater*abs(tempWater - tempWall)*(.914)^3/(nuWater*alphaWater);
%raleigh number calc
   raWall2 = g*betaWater*abs(tempWater - tempWall)*(1.557528)^3/(nuWater*alphaWater);
   raWall3 = g*betaWater*abs(tempWater - tempWall) * (2.590)^3/(nuWater*alphaWater);
   raWall4 = g*betaWater*abs(tempWater - tempWall)*(1.557528)^3/(nuWater*alphaWater);
   raWall5 = g*betaWater*abs(tempWater - tempWall) * (.3048)^3/(nuWater*alphaWater); %
There are two wall 5s
   raWall6 = g*betaWater*abs(tempWater - tempWall) * (.3048)^3/(nuWater*alphaWater);



   %Laminar Flow
   charLengthBottom = areaBottom/(12.8*2+5.49*2);
   raBottomSurface =
g*cos(7.82)*betaWater*abs(tempWater-tempWall)*charLengthBottom^3/(nuWater*alphaWater);
%nusselt number calcs
   nuWall1 = (.680+.670*raWall1^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));
   nuWall2 = (.680+.670*raWall2^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));
   nuWall3 = (.680+.670*raWall3^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));
   nuWall4 = (.680+.670*raWall4^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));
   nuWall5 = (.680+.670*raWall5^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));
   nuWall6 = (.680+.670*raWall6^(1/4)/(1+(.492/prWater)^(9/16))^(4/9));

   nuBottomSurface = (.825+.387*raBottomSurface^(1/6)/(1+(.492/prWater)^(9/16))^(8/27))^2;

   hWater1 = nuWall1*kWater/(.914); %Average Convection Coefficient Calcs
   hWater2 = nuWall2*kWater/(1.557528);
   hWater3 = nuWall3*kWater/(2.590);
   hWater4 = nuWall4*kWater/(1.557528);
```

```
    hWater5 = nuWall5*kWater/(.3048);
    hWater6 = nuWall6*kWater/(.3048);

    hWaterBottomSurface = nuBottomSurface*kWater/(charLengthBottom);

    hWallTotal =
hWater1+hWater2+hWater3+hWater4+hWater5+hWater6+hWaterBottomSurface;
    reTop = u*12.8/nuAir; % assume Laminar
    nuTop = 0.664*reTop^.5*prAir^(1/3);
    hTop = nuTop*kAir/12.8;

    if time == 7 %Setting sunrise at 7am and sunset at 7pm for our model - this will adjust our
daytime and nighttime cycles
        tempSky = tempSkyArray(2);
    elseif time == 19 % 7pm sunset
        tempSky = tempSkyArray(1);
    end

    if time == 25 %"restarts" array
        time = 1;
    end
    qRad = epsilonWater*areaSurface*sigma*(T(i-1)^4-tempSky^4); %calculations for heat
fluxes
    qG = absorptivity*G(time)*areaSurface;
    qConv = hTop*areaSurface*(T(i-1)-tempAir) -
(T(i-1)-tempWall)*(hWater1*area1+hWater2*area2+hWater3*area3+hWater4*area4+hWater5*a
rea5...
        -(T(i-1)-tempWall)*hWater6*area6 -
(T(i-1)-tempWall)*hWaterBottomSurface*areaBottom);
    qEvap = mDot*latentHeatWater;
    qCond = kConcrete/lConcrete*(area1 + area2 + area3 +area4 + area5 +area6
+areaBottom)*(tempWall-tempSoil);
    if cover == 1
        qEvap = 0;
    end
    if mod(i,3600) == 0
        time = time + 1;
    end
    T(i) =T(i-1) + deltat/(rhoWater*v*cpWater)*(-qRad - qConv - qEvap - qCond + qG); %Euler
Method
```

```
    tempWall = tempWater; %Change values for next loop
    tempWater = T(i);
end
time = 0:1/3600:24*length(tempAirArray); %Plot
plot(time,T - 273,'-r');
xlabel('Time (hr)'); ylabel('Pool Temperature (C)');

%For printing RMSE Times

%printTimes = [6.5,8,9,9.5,11,12,13,14,15,16,18,19,20,21,22,6 + 24,7 + 24, 8.5 + 24,10 +
24,10.5 + 24,11.5 + 24,13.5 + 24,14.5 + 24,16 + 24,17 + 24,19 + 24];
%printTimes = printTimes - printTimes(1);
%for n=1:1:length(printTimes)
    %val = T(printTimes(n)*3600 + 1) - 273;
    %fprintf('%1.2f \n',val);
%end
```